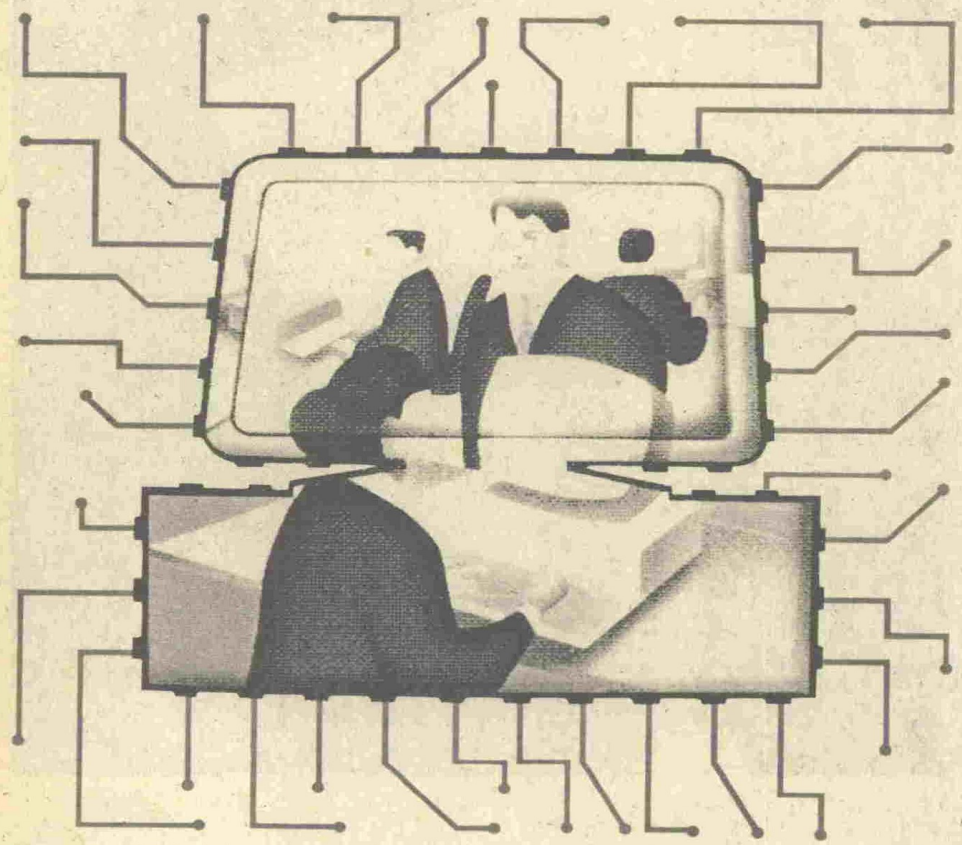
เรื่องน่าอ่าน

สุรวุฒิ สุชินโรจน์

ตำนานและอนาคต สถาปัตยกรรมคอมพิวเตอร์



คอมพิวเตอร์ที่ใช้อยู่ มีจุดกำเนิดอย่างไร และก้าวไปถึงไหนแล้ว

ในรอบสามสิบปีที่ผ่านมานี้ได้มีการออกแบบคอมพิวเตอร์แบบใหม่ ๆ ขึ้นเป็นจำนวนมาก โดยเฉพาะในช่วงสิบปีที่ผ่านมานี้หลังจากเริ่มมีการใช้เทคโนโลยีทาง VLSI ดีไซน์ในการสร้างไมโครโปรเซสเซอร์ การออกแบบคอมพิวเตอร์ได้ก้าวหน้าขึ้นเป็นอันมากจากแบบของคอมพิวเตอร์ที่เคยอยู่บนแผ่นกระดาษในลักษณะของทฤษฎี ได้รับการนำมาผลิตขึ้นเป็นของจริงทั้งที่เป็นการศึกษาและการใช้ประโยชน์จริง ๆ ทั้งนี้เพราะระยะเวลาและต้นทุนในการผลิตได้ลดต่ำลงมาก ทำให้สถาปัตยกรรมคอมพิวเตอร์เหล่านี้แข่งขันกันในด้านความเร็วของการประมวลผลความสะดวกในการเขียนโปรแกรม ความเชื่อถือได้ของการทำงานของระบบรวมทั้งในด้านประโยชน์ใช้สอย แนวความคิดของสถาปัตยกรรมเหล่านี้สามารถแยกออกได้ตามลักษณะเฉพาะเป็นกลุ่ม ๆ ซึ่งได้มีผู้พยายามแยกแยะไว้หลายท่าน ซึ่งจะไม่ขอกล่าวถึงในบทความนี้ แต่จะอธิบายโดยถือตามนัยความคิดเบื้องต้นของความมุ่งหมายของสถาปัตยกรรมคอมพิวเตอร์แต่ละแบบเป็นสำคัญ วิธีนี้จะเป็นประโยชน์สำหรับผู้ที่เกี่ยวข้องกับคอมพิวเตอร์แต่ไม่ได้มีหน้าที่ในการออกแบบคอมพิวเตอร์โดยตรง เพราะจะไม่ติดอยู่กับรายละเอียดทางเทคนิคมากเกินไป

บทความฉบับนี้เป็นการสำรวจแนวความคิดหลากหลายทางสถาปัตยกรรมคอมพิวเตอร์แบ่งออกเป็นกลุ่ม ๆ เพื่อให้มองเห็นความคล้ายหรือความแตกต่างระหว่างแนวความคิดของสถาปัตยกรรมแต่ละกลุ่มในลักษณะกว้าง ๆ ท่านที่มีความสนใจเป็นพิเศษในสถาปัตยกรรมในแนวทางใดแนวทางหนึ่งก็อาจจะสามารถศึกษารายละเอียดเพิ่มเติมได้ในภายหลัง สำหรับรายละเอียดเหล่านี้อาจศึกษาได้จากเอกสารเกี่ยวกับคอมพิวเตอร์ เช่น IEEE Transaction on Computers, ACM Transaction on Computer Systems หนังสือคู่มือคอมพิวเตอร์ของบริษัทต่าง ๆ และเอกสารการประชุมทางคอมพิวเตอร์ (conference proceedings) คำอธิบายพื้นฐานอาจหา

อ่านได้จากหนังสือทางสถาปัตยกรรมคอมพิวเตอร์ที่มีอยู่เป็นจำนวนมาก ซึ่งตัวอย่างของหนังสือบางเล่มได้อ้างอิงไว้ที่ตอนท้ายบทความฉบับนี้

ความเป็นมา

คอมพิวเตอร์ที่รู้จักกันส่วนใหญ่ไม่ว่าจะเป็นไมโครคอมพิวเตอร์ เช่น พีซี จะเป็นค่ายไอบีเอ็ม แอปเปิ้ล คอมมอดอร์ หรือมินิเช่น AS/400 ของไอบีเอ็ม PH3000 ของฮิวเลต-แพคการ์ดหรือเมนเฟรมเช่น DECsystem-10 IBM 3090 เหล่านี้ เป็นแบบของคอมพิวเตอร์ที่ดัดแปลงมาจากสถาปัตยกรรมของคอมพิวเตอร์ที่ได้รับการเสนอไว้ตั้งแต่ปี ค.ศ. 1945 โดยจอห์น ฟอน นอยแมน (John von Neumann) ซึ่งขณะนั้นเขาเป็นที่ปรึกษาในการสร้างคอมพิวเตอร์ยุคแรกที่ชื่อ EDVAC ในสหรัฐอเมริกา แบบของคอมพิวเตอร์ที่เสนอไว้นี้มีส่วนประกอบใหญ่ ๆ สามส่วนคือ ส่วนหน่วยความจำ ส่วนปฏิบัติการคำนวณ (ALU) และส่วนควบคุมการปฏิบัติการ ซึ่งสองส่วนหลังนี้รวมกันเป็นศูนย์การประมวลผลที่เรียกว่าซีพียู (CPU) ตัวซีพียูทำงานโดยอาศัยชุดคำสั่งชุดหนึ่งซึ่งใช้ในการเขียนโปรแกรม ซีพียูจะเรียกคำสั่งของโปรแกรมมาทำงานโดยการอ่านคำสั่งออกมาจากหน่วยความจำมาปฏิบัติทีละคำสั่งต่อเนื่องกันไปตามลำดับในการเก็บ การปฏิบัติคำสั่งแต่ละคำสั่งจะมีสี่ขั้นตอนคือ

1. การอ่านคำสั่งจากหน่วยความจำที่เรียกกันว่าขั้น fetch เข้ามาในซีพียู
2. การตีความคำสั่ง (decoding) ซึ่งเป็นการระบุส่วนคำสั่งปฏิบัติการ (opcode) แยกออกจากส่วนของข้อมูลในคำสั่งนั้น
3. การเคลื่อนย้ายข้อมูลจากหน่วยความจำ (data transfer) เข้ามายังรีจิสเตอร์ในซีพียู
4. การทำการประมวลผลซึ่งเป็นการทำตาม opcode ของคำสั่งนั่นเอง

ทั้งสี่ขั้นนี้รวมกันเรียกว่า execution cycle ซึ่งทำไปทีละรอบรอบละหนึ่งคำสั่ง

การทำงานในลักษณะนี้สามารถใช้งานได้ดีคือ มีความเชื่อมั่นในความถูกต้องของฮาร์ดแวร์ได้สูงเพราะเป็นกลไกง่าย ๆ แต่มีปัญหาด้านความเร็วที่จำกัดและความไม่สะดวกในการเขียนโปรแกรมขนาดใหญ่ ๆ จึงมีการพยายามดัดแปลงเพื่อให้ใช้งานได้ดีขึ้นการดัดแปลงในระยะแรก ๆ เป็นการต่อเติมส่วนการทำงานต่าง ๆ จะยกตัวอย่างการต่อเติมบางอย่างให้เห็นตามลำดับของเวลาการกำเนิดของความคิดนั้น ๆ ดังนี้

1. การใช้ index register คือการเพิ่มรีจิสเตอร์เพื่อใช้ในการระบุตำแหน่งในหน่วยความจำ และเป็นการเพิ่มวิธีระบุตำแหน่งความจำอีกวิธีหนึ่งจากเดิมที่ใช้ตัวเลขระบุตำแหน่งตรง ๆ ซึ่งเรียกว่า "การระบุตำแหน่งความจำแบบ absolute" นับเป็นจุดเริ่มต้นของการใช้ "การระบุตำแหน่งความจำแบบ relative" ซึ่งมีประโยชน์ในการสร้างโมดุลของโปรแกรมที่สามารถเคลื่อนย้ายตำแหน่งได้ในหน่วยความจำ ความคิดนี้เริ่มที่มหาวิทยาลัยแมนเชสเตอร์ ในประเทศอังกฤษปี ค.ศ. 1949 ใช้ครั้งแรกในคอมพิวเตอร์ Datatron ของบริษัท Electro Data Corp.

2. การใช้ program interrupt เริ่มใช้ในเครื่อง Univac ของบริษัท CDC ในปี ค.ศ. 1954 เป็นการเริ่มต้นของโปรแกรมควบคุมระบบที่เริ่มสลับซับซ้อนขึ้น เนื่องจากสามารถเขียนรูทีนที่แทรกเข้ากับการทำงานของโปรแกรมอื่นได้โดยใช้การอินเตอร์รัพต์เป็นส่วนที่สำคัญของเยเนอเรชันแรกของโปรแกรมควบคุมระบบคือ สามารถเก็บส่วนของโปรแกรมควบคุมระบบไว้ในหน่วยความจำหลักได้ไม่ต้องโหลดเข้ามาทุกครั้งที่รันโปรแกรม

3. การใช้ general purpose registers เริ่มในเครื่อง Pegasus ของบริษัท Ferranti LTD. ในปี ค.ศ. 1956 การใช้รีจิสเตอร์ทำให้ความเร็วของการประมวลผลดีขึ้น เพราะการอ่านและบันทึกข้อมูลในรีจิสเตอร์ทำได้รวดเร็วกว่าในหน่วยความจำมาก

4. เริ่มใช้ virtual storage ในคอมพิวเตอร์ Atlas system ของมหาวิทยาลัยแมนเชสเตอร์ ในปี ค.ศ. 1959 วิธีการนี้ทำให้เกิด multiprogramming เป็นจุดกำเนิดของเยเนอเรชันที่สองของโปรแกรมควบคุมระบบซึ่งเป็นสมัยที่เริ่มใช้ระบบ time sharing

การเพิ่มประสิทธิภาพของสถาปัตยกรรมฟอน นอยแมน ที่สำคัญในปัจจุบันคือแนวความคิดที่รู้จักกันในนาม Reduced Instruction Set Computer (RISC) แนวความคิดนี้เริ่มขึ้นในเครื่อง IBM 801 หลังปี ค.ศ. 1970 โดยการออกแบบให้ชุดของคำสั่งมีแต่คำสั่งที่จำเป็นต้องใช้บ่อยที่สุดเท่านั้น คำสั่งเหล่านี้จะถูกทำให้มีจำนวนซีพียูไซเคิลน้อยที่สุด เพื่อให้สามารถปฏิบัติได้อย่างรวดเร็ว ในบางดีไซน์กำหนดให้มีแต่คำสั่งที่ทำงานในหนึ่งซีพียูไซเคิลเท่านั้น เช่น ไมโครโปรเซสเซอร์ SU-MIPS ของ จอห์น เฮนเนสซี จากมหาวิทยาลัยสแตนฟอร์ด แนวความคิดแบบ RISC นี้เมื่อใช้ร่วมกับเทคนิคอื่น ๆ เช่น การทำไปป์ไลน์ทำให้ได้โปรเซสเซอร์ที่มีความเร็วสูง เป็นที่นิยมใช้ในคอมพิวเตอร์ขนาดเล็กที่ต้องการความเร็ว เช่นในเวอร์กสเตชัน

ตัวอย่างของคอมพิวเตอร์ที่ใช้แนวความคิดนี้ในท้องตลาดมีอยู่เป็นอันมาก เช่น เครื่องมินิ HP 930, 950 ของฮิวเลตแพคการ์ด เครื่อง Pyramid 90x ของ Pyramid Technology Corp. และ เวอร์กสเตชัน Ridge 32s ของบริษัท Ridge Computer, Inc.

แนวทางการพัฒนาสถาปัตยกรรมคอมพิวเตอร์ที่ถือว่าเปลี่ยนแปลงไปจากของเดิมอย่างมาก เริ่มจากประมาณปี ค.ศ. 1965 ได้แก่การเริ่มใช้การประมวลผลแบบขนานในเครื่อง CDC star-100, Texas Instrument ASC และ Burroughs ILLIAC IV แนวทางนี้ถือเป็นแนวทางใหม่เพราะเป็นการออกแบบที่พยายามหนีจากลักษณะของสถาปัตยกรรมแบบฟอน นอยแมน แนวทางนี้มีอยู่หลายกระแสซึ่งจะอธิบายได้ดังนี้

สถาปัตยกรรมแบบไม่เป็นฟอน นอยแมน

แบบของการเปลี่ยนแปลงในทางใหม่ ๆ ทางสถาปัตยกรรมคอมพิวเตอร์อาจจะแยกออกได้เป็นสี่แนวทางไม่เรียงตามลำดับเวลา คือ

1. การประมวลผลแบบขนาน (Parallel processing) คือพยายามหาวิธีให้สามารถทำปฏิบัติการของคำสั่งได้พร้อม ๆ กันทีละหลายคำสั่งแทนที่จะทำคำสั่งหนึ่งเสร็จก่อนแล้วค่อยเริ่มคำสั่งต่อไป

2. การออกแบบให้คอมพิวเตอร์มีลักษณะการทำงานใกล้เคียงกับการประมวลผลในภาษาระดับสูง เป็นการพยายามสร้างคอมพิวเตอร์ที่เรียกว่า High Level Language Machine (HLL machine)

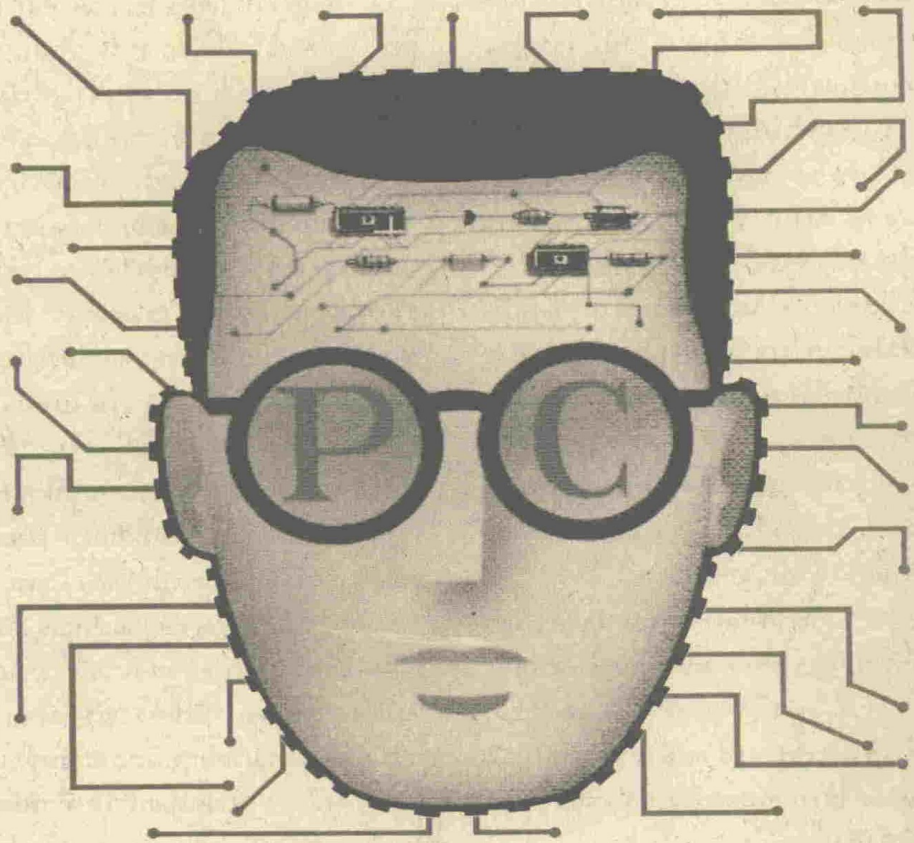
3. การออกแบบคอมพิวเตอร์ให้สอดคล้องกับลักษณะของงานโดยตรง เช่น ในโปรแกรมทางธุรกิจขนาดใหญ่มักมีส่วนเกี่ยวพันกับการใช้ระบบฐานข้อมูลเป็นส่วนใหญ่ก็จะออกแบบคอมพิวเตอร์ที่เอื้ออำนวยในการใช้ระบบฐานข้อมูลเป็นพิเศษคือ เครื่องดาต้าเบส (Database machine)

4. การออกแบบให้เครื่องทำงานด้วยกลไกการคำนวณแบบใหม่ เช่น กลไกการคำนวณที่เรียกว่า dataflow approach หรือ reduction approach เป็นการสลัดทิ้งทั้งแบบของฮาร์ดแวร์และซอฟต์แวร์ของฟอน นอยแมน อย่างค่อนข้างจะสิ้นเชิง

ในที่นี้จะขออธิบายแนวทางเหล่านี้ไปตามลำดับ แต่จะไม่ตามเวลาของการกำเนิดของความคิดแต่ละอย่างเพราะมีเวลาคาบเกี่ยวกันมาก และพัฒนาการส่วนใหญ่เกิดขึ้นในช่วงสามสิบปีนี่เอง

สถาปัตยกรรมแบบขนาน

แนวความคิดหลักของสถาปัตยกรรมแบบนี้ มุ่งความเร็วของคอมพิวเตอร์เป็นหลักเกิดมาจากความคิดที่ว่ากิจกรรมหลายอย่างของโปรเซสเซอร์สามารถทำได้พร้อม ๆ กันไปในเวลาเดียวกัน ซึ่งจะช่วยให้โปรเซสเซอร์ทำงานได้เร็วขึ้นเหมือนกับหลายคนช่วยกันทำงานชิ้นเดียวย่อมเสร็จเร็วกว่าคนใดคนหนึ่งทำแต่เพียงคนเดียว ดังนั้นจึงมีการหาวิธีการหลาย ๆ อย่างเพื่อนำมาปรับปรุงการทำงานของโปรเซสเซอร์ในส่วนต่าง ๆ ให้สามารถทำงานแยกออกจากกันได้ไม่ต้องรอซึ่งกันและกัน วิธีการ

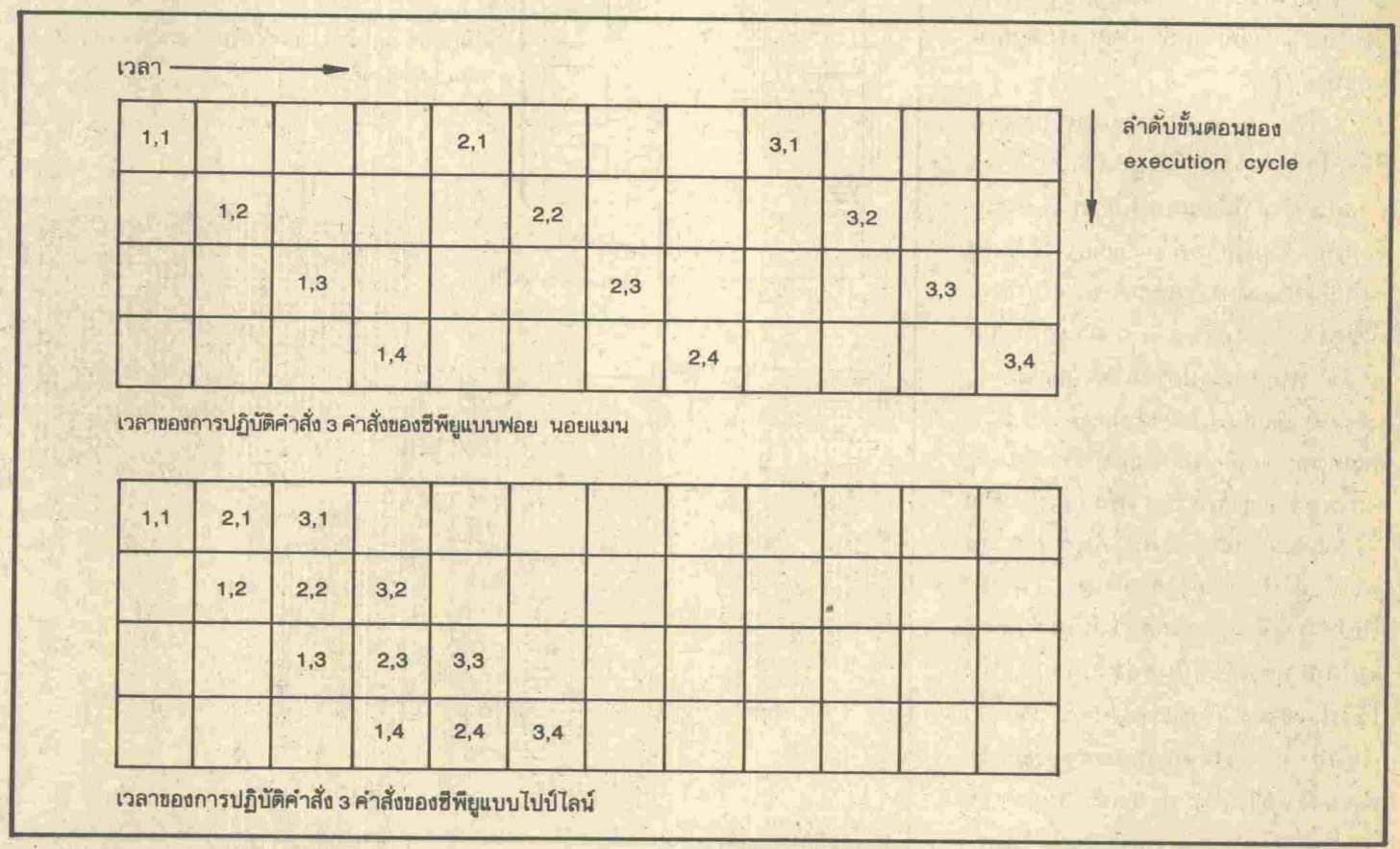


เหล่านี้มีทั้งที่ช่วยกันทำคำสั่งเดียวโดยหลาย ๆ ฝ่ายหรือทำโปรแกรมเดียวกัน แต่แยกเป็นหลาย ๆ ส่วน หรือกระทั่งทำงานแยกกันตามโปรเซส (process) ซึ่งเป็นการทำที่แยกกันในระดับสูงขึ้นไป

วิธีการประมวลผลแบบขนานเป็นแบบที่นิยมใช้มากที่สุดในการเพิ่มความเร็วให้กับโปรเซสเซอร์ คอมพิวเตอร์สมัยใหม่ที่เราใช้กันอยู่ส่วนใหญ่เช่น เครื่อง AS/400 ไอบีเอ็มพีซี เอที อาศัยวิธีที่ใช้กันโดยทั่วไปคือการนำเทคนิคการประมวลผลแบบขนานบางอย่างมาใช้ในดีไซน์ เช่น เทคนิคการทำไปป์ไลน์ ตัวอย่างเช่น การทำไปป์ไลน์ของคำสั่ง ซึ่งทำได้โดยการแยกส่วนการทำงานของส่วนควบคุมการปฏิบัติการ (control unit) ออกเป็นส่วน ๆ ให้ทำงานเป็นอิสระจากกันคือ ส่วนหนึ่งทำการอ่านคำสั่งจากหน่วยความจำหลัก อีกส่วนทำการตีความคำสั่ง อีกส่วนทำการเคลื่อนข้อมูลให้ทำพร้อมกันได้ในแต่ละหน่วยเวลา (หนึ่ง clock pulse) โดยวิธีนี้เครื่องก็จะสามารถปฏิบัติคำสั่งได้เร็วขึ้น แทนที่จะต้องรอทำแต่ละคำสั่งให้เสร็จก่อนแล้วจึงจะขึ้นคำสั่งต่อไป ก็ทำแต่ละขั้นตอนใน execution cycle ไปพร้อมกันโดยให้แต่ละขั้นทำกันคนละคำสั่งเลื่อนส่งกันต่อ ๆ ไปเป็นลำดับ คือเมื่อคำสั่งแรกทำขั้นตอนที่หนึ่งใน execution cycle ซึ่งเมื่อเสร็จแล้วก็จะเลื่อนไปทำขั้นตอนที่สอง ขณะที่เริ่มทำขั้นตอนที่สองนี้คำสั่งถัดไปก็เริ่มทำขั้นที่หนึ่งไปเลย ดังนั้นเมื่อคำสั่งที่หนึ่งทำอยู่ในขั้นที่สามคำสั่งที่สองจะอยู่ในขั้นที่สอง และคำสั่งที่สามซึ่งเป็นคำสั่งใหม่จะอยู่ในขั้นที่หนึ่งเป็นอย่างนี้ต่อ ๆ ไป จะเห็นได้ว่าวิธีนี้จะประหยัดเวลาการปฏิบัติงานได้ครึ่งหนึ่งหรือเร็วขึ้นเท่าตัว ดังจะเห็นได้จากการเปรียบเทียบดังรูปที่ 1

เทคนิคอีกอย่างหนึ่งคือการปฏิบัติคำสั่งเดียวกับข้อมูลหลาย ๆ ชิ้น เหมือนการทำบวกหรือลบอะเรย์สองอะเรย์เข้าด้วยกัน แต่ทำแบบทีเดียวทั้งอะเรย์ เทคนิคแบบนี้เรียกว่า vector processing

เทคนิคการประมวลผลเหล่านี้มักใช้ร่วมกับเทคนิคอื่น ๆ เช่นการใช้บัสหลายชุดระหว่างหน่วยความจำกับซีพียู (many memory buses) เพื่อให้



รูปที่ 1 แสดงการเปรียบเทียบเวลาที่ใช้ในการปฏิบัติคำสั่งของซีพียูแบบฟอน นอยแมนกับแบบไปป์ไลน์ ตัวเลขที่เป็นคู่เช่น 1, 2 เลขหน้าบอกว่าเป็นคำสั่งที่ 1 เลขหลังบอกว่าเป็นขั้นตอนที่ 2 ใน execution cycle

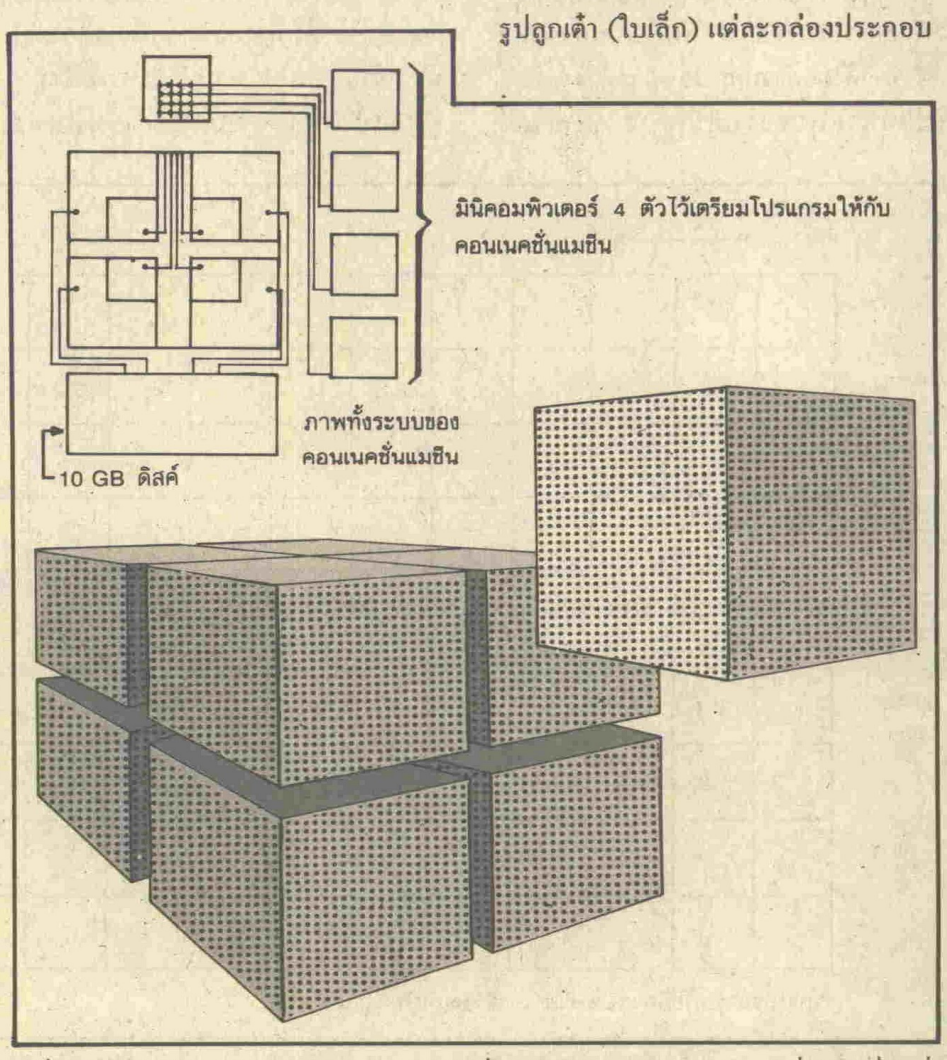
การอ่านข้อมูลและคำสั่งทำได้เร็วขึ้นหรือการใช้แคช (cache memory) ซึ่งเป็นหน่วยความจำขนาดเล็กอยู่ระหว่างหน่วยความจำหลักกับซีพียู รวมทั้งการใช้โคโปรเซสเซอร์ซึ่งเป็นการต่อโปรเซสเซอร์แบบพิเศษพ่วงเข้าไปกับหน่วยควบคุมการปฏิบัติการและหน่วยความจำเพื่อช่วยในการประมวลผลบางอย่างของซีพียู เช่น การต่อพ่วง array processor AP-120B ของบริษัท Floating Point System เข้ากับเครื่องมินิคอมพิวเตอร์หรือการต่อโคโปรเซสเซอร์ 80287 เข้ากับเครื่องพีซีเอทีโปรเซสเซอร์เฉพาะเหล่านี้บางอย่างก็ได้พัฒนาขึ้นจนเป็นคอมพิวเตอร์แบบพิเศษของตัวเอง ตัวอย่างเช่น Systoric array ของเอช ที คุง ที่ใช้ในงานเรียลไทม์อิมเมจโปรเซสซิ่ง ซึ่งเป็นส่วนที่วิเคราะห์ข้อมูลการมองเห็นของหุ่นยนต์

วิธีการประมวลผลแบบขนานอีกวิธีหนึ่งคือ การต่อซีพียูหลาย ๆ ตัวเข้าด้วยกัน ซึ่งทำได้ทั้งแบบที่ใช้หน่วยความจำหลักร่วมกันเรียกว่า tightly-coupled multiprocessors และแบบที่แยกออกเป็นคอมพิวเตอร์หลาย ๆ ตัวต่อกัน แต่ละตัวมีหน่วยความจำของตัวเองเรียกว่า loosely-coupled multiprocessors หรือ multicomputer โปรเซสเซอร์หรือคอมพิวเตอร์เหล่านี้สามารถสร้างขึ้นเป็น VLST ชิพ แต่ละชิพมีได้หลาย ๆ ตัว ทำให้ได้โปรเซสเซอร์ราคาถูก ดังนั้นในปัจจุบันจึงสามารถสร้างคอมพิวเตอร์แบบหลายซีพียูได้เป็นสองแบบ แบบหนึ่งใช้โปรเซสเซอร์ขนาดใหญ่ ๆ ไม่กี่ตัว เรียกว่าเป็นการจัดแบบ large grain ซึ่งพัฒนาออกไปเป็นเครื่องเมนเฟรม เช่น IBM 4341 PO2 หรือเป็นซูเปอร์คอมพิวเตอร์เช่น Cray X-MP ของบริษัท Cray Research และอีกแบบใช้โปรเซสเซอร์ขนาดเล็ก ๆ จำนวนมากเรียกว่าจัดแบบ small grain โดยเฉพาะอย่างยิ่งการจัดแบบ small grain นี้ นำไปสู่ความคิดของการประมวลผล "แบบขนานอย่างมาก" ที่เรียกว่า massively parallel processing เช่นเครื่อง connection machine ของแดนนี่ ฮิลลิส ผู้ตั้งบริษัท Thinking Machines Corp. หรือเครื่อง Cosmic Cube ของมหาวิทยาลัย Caltech

การนำโปรเซสเซอร์หลาย ๆ ตัวมาต่อกันนี้เป็นสิ่งที่ได้รับความสนใจในหมู่นักออกแบบคอมพิวเตอร์มาก เพราะมีปัญหาการสื่อสารระหว่างโปรเซสเซอร์ที่ชวนให้คิด มีรูปแบบการต่อที่น่าสนใจอยู่หลายรูปแบบ ตัวอย่างเช่น รูปแบบที่เรียกว่าไฮเปอร์คิวบ์ (hyper cube) ทรีแมชีน (tree machine) หรือคอนเน็คชั่นแมชีน จะขอยกตัวอย่างของคอนเน็คชั่นแมชีนให้เห็นลักษณะของวิธีการและปัญหาพอสังเขป

Connection Machine

มองจากภายนอกเครื่องคอนเน็คชั่นแมชีน เป็นรูปกล่องที่ประกอบด้วยกล่องรูปลูกเต๋า 8 ลูกต่อกันเป็นลูกเต๋าใบใหญ่ ดังที่แสดงในรูปที่ 2 ระหว่างลูกมีช่องว่างเว้นที่ไว้สำหรับระบายอากาศ กล่องรูปลูกเต๋า (ใบเล็ก) แต่ละกล่องประกอบ

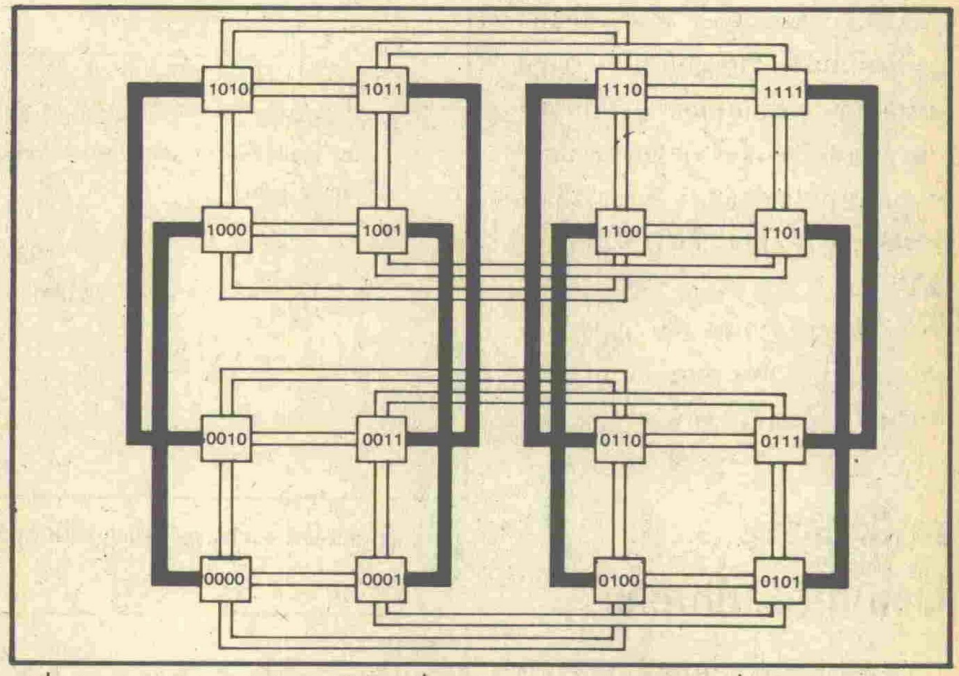


รูปที่ 2 ไดอะแกรมข้างบนแสดงการจัดระบบของคอนเนคชั่นแมชีน และรูปล่างแสดงคอนเนคชั่นแมชีนที่ยกหนึ่งในแปดส่วนออกมาให้ดูทางขวามือ

ด้วยโปรเซสเซอร์การ์ด 16 แผ่น แต่ละแผ่นมีโปรเซสเซอร์ชิพ 32 ตัว แต่ละชิพมีโปรเซสเซอร์อยู่ 16 ตัว โดยที่โปรเซสเซอร์แต่ละตัวนี้มีหน่วยความจำของตัวเองคือเป็นคอมพิวเตอร์เล็ก ๆ ตัวหนึ่งรวมเป็นคอมพิวเตอร์เล็ก ๆ เหล่านี้ต่อกันอยู่ทั้งหมด 65,536 ตัว เวลารันโปรแกรมผู้เขียนโปรแกรมต้องระบุแยกส่วนของปัญหาออกเป็นส่วนย่อย ๆ เพื่อให้แต่ละส่วนทำงานไปได้พร้อม ๆ กัน แล้วจึงป้อนให้กับคอมพิวเตอร์อีกตัวหนึ่งที่ทำหน้าที่เตรียมโปรแกรมให้กับตัวเครื่องของคอนเน็คชั่นแมชีนจริงอีกที จะเห็นได้จากในรูปที่ 2 ดังนั้นการใช้คอมพิวเตอร์ประเภทนี้จึงมีความยุ่งยากกว่าที่เราใช้คอมพิวเตอร์ทั่วไปไม่น้อย

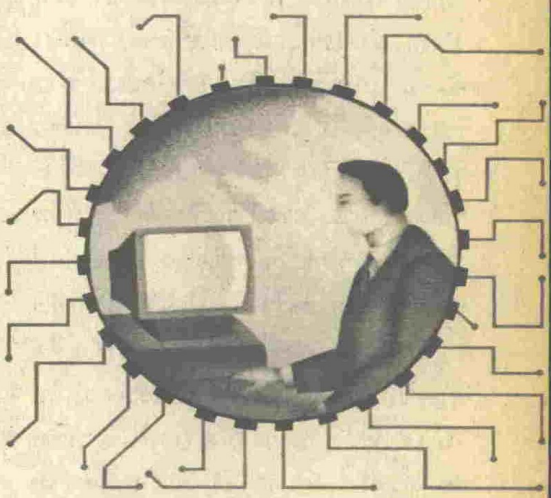
โปรเซสเซอร์แต่ละตัวในคอนเน็คชั่นแมชีนสามารถสื่อสารถึงกันได้โดยผ่านทางวงจรที่เรียกว่า router ซึ่งต่อโยงกันเป็นเครือข่ายดังแสดงในรูปที่ 3 เครือข่ายนี้จะไม่ต่อโยงโดยตรงจากโปรเซสเซอร์ตัวหนึ่งไปยังทุกตัว เนื่องจากจะทำให้การต่อสายมีจำนวนมหาศาล (กว่าสองพันล้านสาย คือ [n(n-3)/2+n]) แต่จะต่อกันในลักษณะพิเศษดังที่เห็นในภาพ ซึ่งโดยวิธีนี้ทำให้เส้นทางการต่อโยงระหว่างโปรเซสเซอร์มีจำนวนประมาณ 25,000 สายเท่านั้น

โปรเซสเซอร์แต่ละตัวจะต่ออยู่กับหนึ่ง router ฉะนั้นปัญหาจึงมีว่าจะต่อ router เหล่านี้เข้าด้วยกันได้อย่างไรจึงจะสามารถสื่อสารถึงกันจาก router หนึ่งไปยังอีกตัวหนึ่งได้ ดังตัวอย่างในรูปที่ 3 ที่แสดงการต่อ router 16 ตัวเข้าด้วยกัน สมมติว่า router ที่ตำแหน่ง 1000 (คือ address เป็นเลขฐานสองจำนวนสี่บิต) ต้องการส่งข้อมูลไปยัง router 0100 จะเห็นว่า router 1000 ต่ออยู่กับ router ตัวที่ 1010, 1001, 1100 และ 0000 ตัว router 1000 จะเปรียบเทียบบิตแรกของเป้าหมายคือ 0100 ซึ่งได้แก่ 0 กับบิตแรกของ router ที่ต่อกันอยู่ (คือสี่ตัวที่ต่อกับ 1000) ซึ่งจะเลือกเอา 0000 เป็นสถานีต่อไป ข้อมูลก็ถูกส่งมายัง 0000 เพื่อส่งต่อไปอีก ขั้นต่อไปจะเห็นว่า 0000 ต่ออยู่กับ 0010, 1000, 0001 และ 0100 ตัว router 0000 ก็จะเปรียบเทียบบิตที่สองของเป้าหมายคือ 1 กับบิตที่สองของ router ที่ต่ออยู่กับมันในที่นี้ก็จะเลือกได้ 0100 ซึ่งเป็นเป้าหมายพอดี



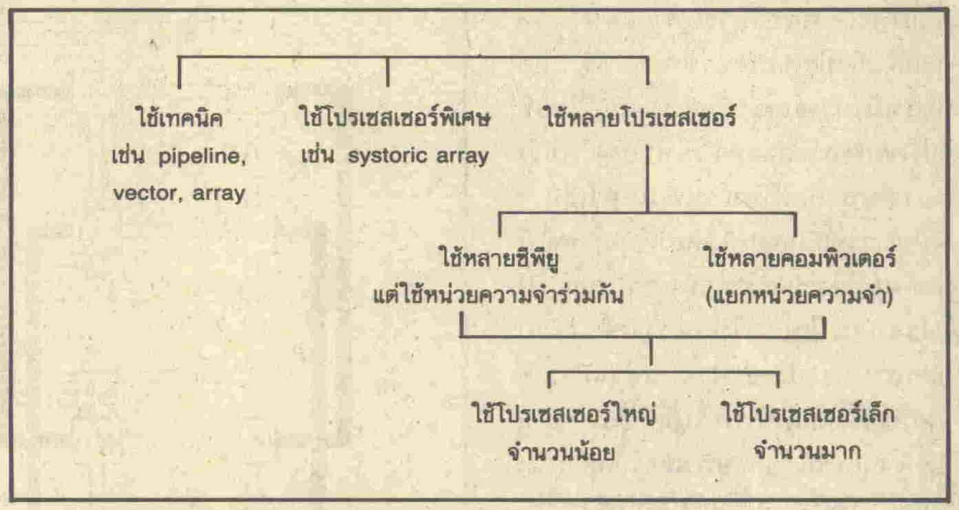
รูปที่ 3 แสดงการต่อโปรเซสเซอร์ 16 ตัวเข้าด้วยกันสื่อสารถึงกันผ่านทาง router ตัวเลขที่เห็นระบุตำแหน่ง (address) ของโปรเซสเซอร์แต่ละตัว

จากตัวอย่างที่ยกมานี้จะเห็นว่าการประมวลผลแบบขนานแบบหลายซีพียูมีปัญหาใหญ่ ๆ สองประการคือ หนึ่งการเตรียมโปรแกรมเพื่อใช้กับเครื่องสองการจัดเครือข่ายการสื่อสารระหว่างซีพียูคอมพิวเตอร์ประเภทนี้จึงนิยมใช้ในงานที่ต้องคำนวณเลขด้วยความเร็วสูงเท่านั้นเช่นในงานสร้าง graphic simulation ขนาดใหญ่ ๆ หรือในการคำนวณข้อมูลจากดาวเทียมเพื่อการพยากรณ์อากาศซึ่งต้องการผลอย่างรวดเร็ว ปัญหาของโปรแกรมคอมพิวเตอร์แบบขนานเป็นปัญหาที่น่าสนใจมากเป็นพิเศษในปัจจุบันเนื่องจากไปสอดคล้องกับพัฒนาการทางภาษาคอมพิวเตอร์ประเภทที่เรียกว่า



descriptive languages ซึ่งเป็นปัญหาการวิจัยที่กำลังได้รับความสนใจอย่างสูง ภาษาคอมพิวเตอร์แบบนี้เป็นรูปแบบของภาษาที่พยายามหนีข้อจำกัดของสถาปัตยกรรมแบบฟอน นอยแมน ซึ่งประเด็นนี้จะมองเห็นได้ชัดเจนขึ้นในคำอธิบายต่อไป

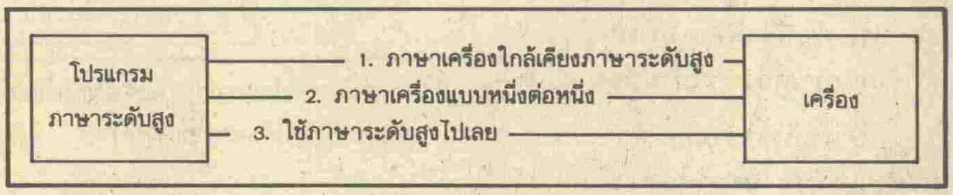
โดยสรุปการพัฒนาทางการประมวลผลแบบขนานอาจจะเขียนออกมาเป็นไดอะแกรมได้ดังรูปที่ 4



รูปที่ 4 แสดงสถาปัตยกรรมหลายแบบตามแนวทางการประมวลผลแบบขนาน

สถาปัตยกรรมที่ถือตามภาษาเป็นหลัก

สถาปัตยกรรมคอมพิวเตอร์อีกแนวทางหนึ่งเป็นที่ทราบกันในนาม High Level Language Computer Architecture เกิดจากแนวความคิดที่ว่าความแตกต่างระหว่างสังกัปของภาษาคอมพิวเตอร์ เช่น ภาษาปาสคาล ภาษาโคบอล ฯลฯ กับสังกัปของการทำงานของตัวเครื่องมีระยะห่างกันมาก ตัวอย่างเช่น ในภาษาปาสคาลจะมี block structure ในรูป BEGIN...END ซึ่งช่วยกำหนดกฎเกณฑ์เกี่ยวกับขอบเขตของตัวแปร (scope rules) ในโปรแกรม แต่เราจะไม่พบกลไกแบบนี้ในสถาปัตยกรรมของคอมพิวเตอร์ทั่วไปเช่นเครื่องคอมพิวเตอร์เอที พีซี หรือการใช้อะเรย์ การใช้เรคอร์ดในภาษาระดับสูงซึ่งไม่พบอยู่ในสถาปัตยกรรมของเครื่องคอมพิวเตอร์ทั่วไปช่องว่างนี้ไม่เพียงแต่จะเกิดขึ้นระหว่างภาษาและเครื่องเท่านั้น แต่ปรากฏอยู่ระหว่างโปรแกรมควบคุมระบบกับเครื่องด้วยที่นับว่าสำคัญคือสถาปัตยกรรมคอมพิวเตอร์ทั่ว ๆ ไปไม่มีสิ่งที่ช่วยในการจัดการกับโปรเซสเลย ทั้งที่การจัดการโปรเซสเป็นกิจกรรมสำคัญที่ต้องปฏิบัติบ่อยที่สุดอย่างหนึ่งในโปรแกรมควบคุมระบบแบบมัลติทาสกิ้ง (multitasking) การใช้เซมาฟอร์ (semaphore) คริติคอลเซกชัน (critical section) และมอนิเตอร์ (monitor) เหล่านี้ทำในซอฟต์แวร์ทั้งสิ้น



รูปที่ 5 แสดงแนวความคิดที่ต่างกันสามแบบในการออกแบบเพื่อลดช่องว่างระหว่างภาษาระดับสูงกับเครื่อง

ปัญหาช่องว่างเหล่านี้ทำให้เกิดผลเสียบางอย่างเช่น ทำให้โปรแกรมคอมพิวเตอร์ใหญ่เกินกว่าที่ควรจะเป็น ดังจะเห็นได้จากคำสั่งในภาษาระดับสูงหนึ่งคำสั่งอาจจะแปลเป็นคำสั่งในภาษาเครื่องเป็นจำนวนมาก ๆ ขอยกตัวอย่างคำสั่งในภาษา PL/I:

```
C(I,J) = A(I,J) + B(J,I);
```

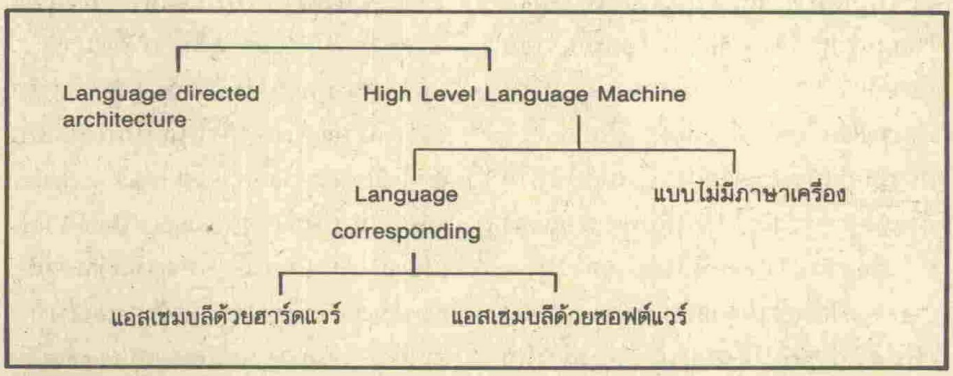
คำสั่งนี้อาจถอดเป็นภาษาเครื่องบนไอบีเอ็ม system 370 ได้ถึง 75 คำสั่ง เนื่องจากต้องมีรหัสที่ใช้ในการเช็คขอบเขตของอะเรย์ ซึ่งถ้ามีคำสั่งในภาษาเครื่องที่ใช้เช็คขอบเขตของอะเรย์โดยตรง เช่น คำสั่ง BOUNDS-CHECK ใน Motorola 68000 ภาษาเครื่องของคำสั่งนี้ก็จะสั้นลงอีกเยอะโปรแกรมก็จะทำงานได้เร็วขึ้นประหยัดเนื้อที่ในหน่วยความจำหลักและทำให้การเขียนคอมพายเลอร์ง่ายขึ้นอีกด้วย

ด้วยเหตุนี้จึงเกิดมีแนวความคิดของสถาปัตยกรรมคอมพิวเตอร์ที่ทำให้ช่องว่างเหล่านี้แคบลง แนวความคิดนี้แสดงได้เป็นไดอะแกรมดังรูปที่ 5

แบบแรกคือลักษณะของสถาปัตยกรรมคอมพิวเตอร์ที่เรียกว่า Language directed architecture เป็นการออกแบบที่มีความมุ่งหมายให้ใกล้เคียงกับภาษาระดับสูงบางจำพวก เช่น เครื่อง Burroughs B3500 สำหรับภาษาโคบอล

ปัจจุบันบริษัท Burroughs รวมกับบริษัท Sperry Univac กลายเป็นบริษัท UNISYS บริษัท Burroughs เก่าเป็นผู้บุกเบิกสถาปัตยกรรมคอมพิวเตอร์แบบนี้ที่สำคัญบริษัทหนึ่ง Intel iAPX 432 สำหรับภาษา Ada ไมโครโปรเซสเซอร์ของบริษัท national Semiconductor ซีรีส์ 32000 และของบริษัท Motorola MC68020 สำหรับภาษาที่ใช้ structured programming และไมโครโปรเซสเซอร์ SOAR สำหรับภาษา Smalltalk ของมหาวิทยาลัยแห่งแคลิฟอร์เนียเมืองเบอร์คเลย์ เป็นต้น อีกสองแบบคือแบบที่สองและที่สามเรียกว่า High Level Language Machine เป็นสถาปัตยกรรมสำหรับภาษาระดับสูง โดยเฉพาะแบบที่สองนั้นมีภาษาเครื่องที่ตรงกับคำสั่งในภาษาระดับสูงแบบหนึ่งต่อหนึ่ง ตัวอย่างเช่น Symbolic Machine ของบริษัท Texas Instrument สำหรับภาษา LISP สถาปัตยกรรมแบบนี้มีชื่อเรียกกันโดยเฉพาะว่า Language corresponding architecture สร้างได้สองวิธี วิธีหนึ่งใช้เครื่องแอสเซมบลีภาษาระดับสูงก่อนแล้วจึงรันภาษาเครื่องที่ได้ อีกวิธีหนึ่งใช้ซอฟต์แวร์แอสเซมบลี ส่วนแบบสุดท้ายที่ใช้ภาษาระดับสูงเป็นภาษาเครื่องมีชื่อเรียกว่า direct execution architecture เป็นแบบที่ไม่เป็นที่นิยม เนื่องจากมีข้อจำกัดหลายประการ เป็นต้นว่า ถ้าหากโปรแกรมดีบักผู้ใช้จะไม่สามารถทราบได้จนกว่าจะรันโปรแกรมไปแล้ว ซึ่งอาจกินเวลานานกว่าจะรู้ไม่เหมือนกับการแอสเซมบลีหรือคอมพายล์ก่อน ตัวอย่างของเครื่องแบบนี้ เช่น LAX2 machine ของ Royal Institute of Technology ในประเทศสวีเดน สำหรับภาษา LAX

วิธีสร้างคอมพิวเตอร์ตามสถาปัตยกรรมในแนวทางนี้คือ การสร้างให้คอมพิวเตอร์มีคำสั่งตามลักษณะคำสั่งในภาษาระดับสูงมีวิธีที่ช่วยในการจัดการกับ data structure จำพวกอะเรย์ หรือเรคอร์ดแยกคำสั่งกับข้อมูลออกจากกันในหน่วยความจำโดยการใช้ tag คือให้มีบิตที่ระบุว่าเวิร์ด (word) แต่ละเวิร์ดนั้นเป็นคำสั่งหรือข้อมูลถ้าเป็นข้อมูลเป็น type ไหน (บูเลียน โฟลทติ้งพอยท์ ฯลฯ) ออกแบบให้มีการจัดการโมดุลที่ระดับภาษาเครื่อง มีการใช้ capability-based addressing คือการมองโมดุลไฟล์ กลุ่มของตัวแปรหรืออื่น ๆ ที่เกี่ยวกับการใช้หน่วยความจำเป็นออปเจ็คต์ที่สามารถระบุสิทธิในการอ่าน เขียน หรือทำลายได้ในการอ้างอิงระหว่างกัน มีการจัดการกับโปรเซส



รูปที่ 6 แสดงสถาปัตยกรรมหลายแบบตามแนวทาง High Level Language Architecture

เราอาจเขียนไดอะแกรมแสดงแนวทางของสถาปัตยกรรมที่ถือภาษาเป็นหลักได้ดังรูปที่ 6

สถาปัตยกรรมตามลักษณะงาน

แนวความคิดของสถาปัตยกรรมแบบนี้มีตัวอย่างที่ดีคือ Database machine สำหรับงานทางธุรกิจ ซึ่งใช้ระบบฐานข้อมูลขนาดใหญ่ ๆ และ Array processor สำหรับงาน image processing ซึ่งปกติมีการคำนวณแต่ละส่วนของภาพแยกเป็นส่วน ๆ เป็นอิสระจากกันสถาปัตยกรรมในลักษณะนี้ทำให้ความเร็วในการใช้งานเฉพาะอย่างนั้นดีขึ้น แต่ขณะเดียวกันก็ลดความเป็นทั่วไปลง เนื่องจากการใช้ระบบฮาร์ดแวร์ที่เป็นการเฉพาะเช่นการใช้ associative memory ในเครื่องดาต้าเบส

สถาปัตยกรรมแบบตามลักษณะงานเลือกเอาลักษณะเฉพาะที่เป็นจุดสำคัญของงานนั้น ๆ มาทำในฮาร์ดแวร์ ตัวอย่างเช่น ในกรณีของเครื่องดาต้าเบสก็สร้างให้สามารถปฏิบัติส่วนต่าง ๆ ของคำสั่งได้ในลักษณะการประมวลผลแบบขนาน เช่น อ่านไฟล์พร้อมกันทีเดียวหลายไฟล์ ปฏิบัติส่วนของคำสั่งหลาย ๆ ส่วนพร้อมกันในการอ่านหรืออัพเดตข้อมูลก็ทำในดิสค์ตรง ๆ เช่น ในการปฏิบัติคำสั่งให้หารายชื่อของพนักงานที่มีเงินเดือนมากกว่าสองหมื่นบาททั้งหมด ซึ่งเป็นรูปคำสั่งทั่วไปที่ใช้บ่อยในดาต้าเบสในเครื่องคอมพิวเตอร์ธรรมดาจะอ่านข้อมูลทั้งหมดเข้าไปเลือกในหน่วยความจำหลักทำให้เสียเวลาในเครื่องดาต้าเบสจะทำให้ในดิสค์เลย นอกจากนี้ในเครื่องคอมพิวเตอร์ธรรมดาการใช้ดาต้าเบสผ่านชั้นของซอฟต์แวร์สลับซับซ้อนหลายชั้นทำให้

การใช้ข้อมูลช้าลงในเครื่องดาต้าเบสจะทำโดยคำสั่งของมันเองตรง ๆ เป็นการทำในฮาร์ดแวร์จึงรวดเร็วกว่ามาก

ตัวอย่างของเครื่องดาต้าเบสได้แก่ เครื่อง RAP ของมหาวิทยาลัยโตรอนโต ประเทศแคนาดา เครื่อง DBC (ย่อมาจาก Database Computer) ของมหาวิทยาลัยโอไฮโอ ประเทศสหรัฐอเมริกา และเครื่อง Britton Lee IDM 500

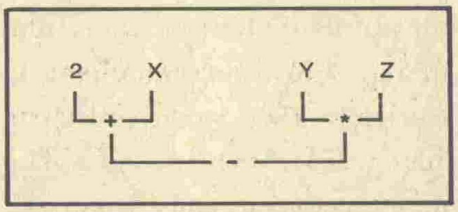
สถาปัตยกรรมตามโมเดลการคำนวณแบบใหม่

บรรดาสถาปัตยกรรมที่กล่าวมาทั้งหมดมีสิ่งที่เหมือนกันอยู่อย่างหนึ่งนั่นคือ ทุกแบบใช้กลไกการคำนวณที่เรียกว่า control-driven model คือมีวิธีการประมวลผลได้ดังนี้

1. การปฏิบัติของคำสั่งเกิดจากการใช้คำสั่ง (opcode) เป็นตัวกระตุ้นให้เกิดปฏิบัติการโดยตรง

2. การส่งผ่านข้อมูลระหว่างคำสั่งเกิดขึ้นโดยทางอ้อมคือ ผ่านการอ้างอิงถึงตำแหน่งความจำเดียวกัน

3. การไหลของคำสั่งกับการไหลของข้อมูลแยกจากกัน โดยทฤษฎีแล้วโมเดลการคำนวณแบบที่ไม่เป็นลักษณะนี้มีอยู่หลายโมเดล แต่ที่เป็นที่สนใจนำมาใช้ในการสร้างคอมพิวเตอร์ ซึ่งได้รับความสนใจเป็นอันมากในปัจจุบันมีอยู่สองโมเดล คือ dataflow approach และ reduction approach ความสำคัญของโมเดลสองแบบนี้คือทำให้การประมวลผลแบบขนานเกิดขึ้นเป็นธรรมชาติ คือ ผู้ใช้ไม่ต้องเตรียมโปรแกรมที่ระบุว่าส่วนใดเป็นอิสระจากกัน เช่น ที่ต้องทำในโปรแกรมของคอมพิวเตอร์แบบขนาน ทั้งนี้โดยการใช้ภาษาชนิดที่เรียกว่า declarative language คือภาษาที่โปรแกรมมีแต่การ declare ว่าอะไรเป็นอะไรตรงข้ามกับภาษาแบบ procedural ที่โปรแกรมคือขั้นตอนที่บอกว่า "ทำอย่างไร"



รูปที่ 7 กราฟของดาต้าโฟลวเพื่อคำนวณ
$F = (2+X) - Y*Z$

ในที่นี้จะอธิบายโมเดลการคำนวณและสถาปัตยกรรมคอมพิวเตอร์แบบดาต้าโฟลวเป็นตัวอย่าง

Dataflow Architecture

โมเดลการคำนวณของดาต้าโฟลวเป็นแบบ data-driven คือความพร้อมของข้อมูลเป็นตัวกระตุ้นให้เกิดปฏิบัติการ ซึ่งแสดงให้เห็นได้ด้วยรูปกราฟดังในรูปที่ 7

จากในภาพจะเห็นว่ามีโหนดเป็นวงกลมที่ภายในบรรจุโอเปอเรเตอร์อยู่สามโหนดมีเส้นโยงระหว่างโหนดเป็นทางแสดงการไหลของดาต้า กราฟที่แสดงนี้แทนการคำนวณ $F = (2 + X) - (Y*Z)$ สัญลักษณ์ X, Y และ Z แทนค่าที่ยังไม่ปรากฏที่อินพุตของโหนด เมื่อค่าที่อินพุตของโหนดพร้อมเมื่อไรการปฏิบัติการก็เกิดขึ้นทันทีผลที่ได้ก็จะไหลต่อไปยังโหนดต่อไปที่ต่อกันอยู่ เช่น เมื่อค่าของ X มาถึง โหนด "+" ก็จะทำงานเพราะอีกค่าหนึ่ง คือค่า "2" พร้อมอยู่แล้ว ผลจากการบวกจะไหลไปที่โหนด "−" ต่อไป ค่าอินพุต 2 และค่า X ก็จะหายไปจากที่อินพุตของโหนด "+" ถ้าเมื่อไรมีค่าเข้ามาที่อินพุตของโหนดนี้อีกปฏิบัติการนี้ก็จะเกิดขึ้นอีกอย่างเดิม ลักษณะเดียวกันนี้เกิดขึ้นกับโหนด "*" ทางขวามือ ผลที่สุดเมื่อปฏิบัติการบวกและคูณเสร็จแล้วก็จะได้อินพุตครบถ้วนที่โหนด "−" ปฏิบัติการสุดท้ายคือ ลบก็จะเกิดขึ้นได้เป็นค่าผลลัพธ์ ในความเป็นจริงจะมีโหนดต่าง ๆ อยู่หลายชนิดบางโหนดใช้ในการตัดสินใจแบบ IF...THEN...ELSE... บางโหนดก็เป็นลอจิกโอเปอเรเตอร์และอื่น ๆ

วิธีการคำนวณแบบดาต้าโฟลวมีลักษณะน่าสนใจดังนี้

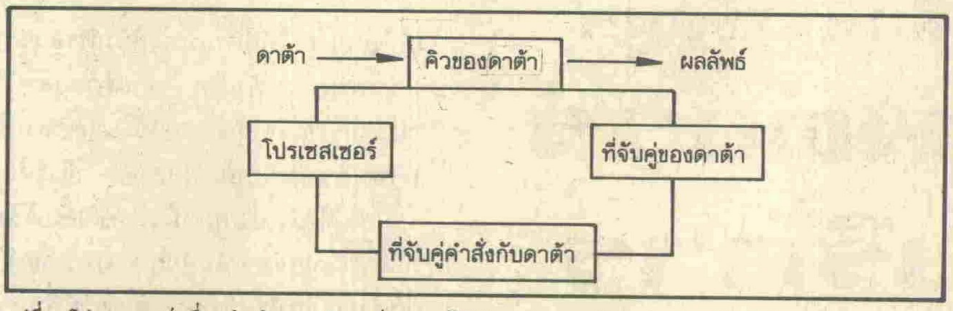
1. ไม่มีตัวแปรในความหมายของตำแหน่งในหน่วยความจำอย่างที่ใช้ในเครื่องแบบฟอน นอยแมน เพราะข้อมูลไหลไปเรื่อย ๆ ไม่ต้องเขียนหรืออ่านจากหน่วยความจำ

2. ปฏิบัติการแต่ละโหนดเป็น local จริง ๆ คือไม่มีการควบคุมจากศูนย์กลาง ดังนั้นจึงทำให้เกิดการประมวลผลแบบขนานโดยธรรมชาติ โหนดไหนพร้อมก็ทำไปเลยหลาย ๆ โหนดปฏิบัติพร้อมกัน

3. ปฏิบัติการแต่ละอย่างไม่มีประวัติคือ ขึ้นกับค่าอินพุตที่ขณะนั้นเท่านั้น คุณลักษณะนี้เรียกว่า referential transparency

4. การทำงานเป็นแบบฟังก์ชันเหมือนกับฟังก์ชันในทางคณิตศาสตร์ ไม่มีผลข้างเคียงทำให้โปรแกรมมีความถูกต้องมากขึ้น การคำนวณแบบดาต้าโฟลวจึงเหมาะกับภาษาแบบฟังก์ชัน ตัวอย่างเช่น ภาษา ลูสิค (LUCID)

สถาปัตยกรรมคอมพิวเตอร์สำหรับการคำนวณแบบดาต้าโฟลวมีหลายแบบ เช่น แบบเซล-บลอค (Cell block architecture) แบบ HDFM (Hughes Data-Flow Machine) และแบบของมหาวิทยาลัยแมนเชสเตอร์ในประเทศอังกฤษ ในที่นี้จะใช้วิธีการทำงานแบบของแมนเชสเตอร์เป็นตัวอย่าง ดังแสดงในรูปที่ 8



รูปที่ 8 โปรเซสเซอร์หนึ่งหน่วยในคอมพิวเตอร์แบบดาต้าโฟลวของแมนเชสเตอร์

จากภาพที่ 8 จะเห็นว่าดาต้าจะเข้ามาอยู่ในคิวก่อน แล้วจะเคลื่อนไปรอจับคู่กัน เพื่อให้ได้ครบชุดสำหรับปฏิบัติการแต่ละอย่าง เช่น ถ้าจะคำนวณ 1+2 ทั้ง 1 และ 2 ก็มาเจอกันที่ที่สำหรับจัดคู่ของดาต้า จากนั้นจึงไหลไปจับคู่กับคำสั่งที่เก็บอยู่ในที่สำหรับจับคู่กับคำสั่ง (คำสั่งที่เก็บอยู่ที่นี่คือคำสั่ง "+" ของ 1+2) เมื่อทั้งคำสั่งและข้อมูลพร้อมแล้วก็เคลื่อนต่อไปปฏิบัติที่โปรเซสเซอร์ ผลที่ได้ก็ไหลต่อไปที่คิวของดาต้า ถ้าเป็นคำตอบก็ออกไปเลย ถ้าไม่ใช่ก็ไหลต่อไปยังที่สำหรับจับคู่ของดาต้าอีกต่อไป วงจรนี้คือหนึ่งหน่วยของโปรเซสเซอร์ตัวคอมพิวเตอร์จริง ๆ อาจสร้างให้มีได้หลาย ๆ หน่วย

ในปัจจุบันนี้ยังไม่ปรากฏว่ามีคอมพิวเตอร์แบบดาต้าโฟลวจำหน่ายในท้องตลาดส่วนใหญ่จะเป็นเครื่องตัวอย่างสำหรับศึกษา ซึ่งการวิจัยทางเครื่องดาต้าโฟลวนี้นับว่าเป็นหัวข้อที่นิยมมาก มีกลุ่มการวิจัยกว่าสามสิบกลุ่มที่กำลังศึกษาเรื่องนี้อยู่ในประเทศญี่ปุ่น อังกฤษ สหรัฐอเมริกา และฝรั่งเศส เครื่องดาต้าโฟลวมีวัตถุประสงค์ให้เป็นคอมพิวเตอร์ขนาดเล็กที่มีความเร็วสูง แต่ที่มหาวิทยาลัยแห่งแคลิฟอร์เนียที่เออร์วีนได้ใช้วิธีดาต้าโฟลวสร้างเครื่องดาต้าเบสราคาของเครื่องตกประมาณ 15,000 ดอลลาร์สหรัฐ จากการทดลองพบว่าในบรรดาเครื่องดาต้าเบสที่ราคาพอ ๆ กัน เครื่องแบบนี้มีประสิทธิภาพสูงกว่าเครื่องที่มีสถาปัตยกรรมแบบฟอน นอยแมนประมาณหกถึงสิบเท่าโดยเฉลี่ย แสดงให้เห็นว่ามีความเป็นไปได้มากที่จะผลิตเครื่องดาต้าโฟลวเป็นสินค้า

สรุป

สถาปัตยกรรมคอมพิวเตอร์ได้รับความสนใจศึกษาอย่างแพร่หลายในปัจจุบัน เนื่องจากสามารถทำได้ค่อนข้างสะดวกกว่าแต่ก่อนมาก วิธีหนึ่งคือสามารถทำได้โดยการจำลองขึ้นเป็นโปรแกรมคอมพิวเตอร์ เพื่อศึกษาคุณลักษณะทางทฤษฎี วิธีนี้เป็นที่นิยมกันมาก อีกวิธีหนึ่งคือการออกแบบเป็น VLSI ชิพที่เป็นเพียงส่วนหนึ่งของระบบจริง ๆ ทำให้ได้ศึกษาจากของจริงโดยไม่ต้องลงทุนมากเกินไป

แนวโน้มของการเปลี่ยนแปลงทางสถาปัตยกรรมคอมพิวเตอร์จากแบบฟอน นอยแมนไปเป็นอย่างอื่นเป็นที่เห็นได้ชัดเจนแล้วในปัจจุบัน การสำรวจแนวทางสถาปัตยกรรมที่เสนอในบทความฉบับนี้ ได้แบ่งแนวทางเหล่านี้เป็นสี่แนวโดยถือตามลักษณะความคิดพื้นฐานของความมุ่งหมายในการออกแบบหรือปรัชญาของสถาปัตยกรรมแต่ละแบบ เพื่อให้มองเห็นความสำคัญของสถาปัตยกรรมนั้น ๆ ในความเป็นจริงคอมพิวเตอร์แต่ละเครื่องเป็นผลจากการพัฒนาที่ผสมผสานเทคนิคและความคิดหลาย ๆ แนวทางเข้าด้วยกัน เพื่อให้ได้ผลที่ดีที่สุดตามแต่จะตั้งเป้าหมายไว้ เช่น ให้ประหยัด ให้มีขนาดเล็ก หรือเพื่อใช้งานหนักไปในทางใดทางหนึ่ง ฯลฯ แต่ความเข้าใจแนวทางหลักของสถาปัตยกรรมคอมพิวเตอร์ในลักษณะที่แสดงนี้จะช่วยให้เรามองความผสมผสานนั้นออกได้

สุดท้าย การเปลี่ยนแปลงทางสถาปัตยกรรมไม่ใช่เป็นแนวทางเดียวที่จะพัฒนาประสิทธิภาพของเครื่องคอมพิวเตอร์ความจริงมีวิธีการอื่น ๆ ที่ใช้ปนกันไปในการออกแบบคอมพิวเตอร์แต่ละเครื่อง เช่นการใช้อุปกรณ์ที่มีประสิทธิภาพดีขึ้น ตัวอย่างเช่น การใช้แกลเลี่ยม อาร์เซไนด์ชิพแทนซิลิคอนชิพ แม้กระทั่งเทคโนโลยีที่ใช้ในการผลิตโปรเซสเซอร์ชิพของคอมพิวเตอร์ก็มีส่วนเกี่ยวข้องกับความจำกัดของประสิทธิภาพของคอมพิวเตอร์ ด้วยเหตุนี้การศึกษาในแง่เฉพาะของสถาปัตยกรรมคอมพิวเตอร์จึงเป็นเพียงส่วนหนึ่งของภาพรวมนี้เท่านั้น

เอกสารอ้างอิง

1. Flynn, M. Some Computer Organization and their Effectiveness IEEE Transaction on Computers C-21(9) pp.948-960 September 1972.
2. Treleaven, Philip C. Decentralised Computer Architecture in New Computer Architecture edited by J. Tiberghien, Academic Press, London, 1984.
3. Kung, H. T. 'Why Systoric Architecture? Computer 15, 37-46, January 1982.
4. Myers, Glenford J. Advances in Computer Architecture John Wiley & Sons, New York, 1982.
5. Milutinovic, Veljoko (editor) High-level Language Computer Architecture Computer Science Press, New York, 1989.
6. Wadge, William W. & Ashcroft, Edward A. Lucid, The Dataflow Language Acdemic Press, London, 1985.